**Student Name: Denzel Tinotenda Pawandiwa**

**Student Number: ST10457186**

**Module: Programing 2B**

**Documentation**

The Contract Monthly Claim System (CMCS) prototype was created to make it easier for Independent Contractor (IC) lecturers to submit their monthly claims, and for coordinators and managers to review and approve them. The idea behind the design is to keep the system simple, clear, and easy to use, while also leaving space for future improvements as the project grows.

**Design Choices and Database Structure**

The database is built around five main entities: **Lecturer, Claim, Coordinator, Manager, and Document**. The Lecturer table keeps basic details like name and contact information. Since a lecturer can submit more than one claim, there’s a one-to-many relationship between Lecturer and Claim. Each claim stores information such as hours worked, hourly rate, total payment, and its status (pending, approved, or rejected). Claims can also have supporting files like timesheets or invoices, so there’s another one-to-many link between Claim and Document.

Coordinators and Managers are kept as separate entities because they play different roles. Coordinators check the claims first, and Managers make the final decision. This step-by-step approval process ensures accountability and makes the system more reliable. The design also makes it easy to add other roles later, like HR or Finance, if the system is expanded.

**GUI Layout**

For the prototype, the user interface was built using **ASP.NET Core MVC**. Each role has its own page so users only see what they need.

* On the **Lecturer page**, there’s a simple form where lecturers can enter the hours they worked, their hourly rate, and any extra notes. They can also upload documents to support their claim.
* The **Coordinator page** lists claims waiting to be reviewed, with Approve and Reject buttons.
* The **Manager page** shows claims that need final approval.
* The **Status page** allows lecturers to check where their claim is in the process.

This layout follows the natural flow: submit → verify → approve → track. By keeping everything separated, it avoids confusion and gives each role a clean and straightforward interface.

**Assumptions**

* Only lecturers who are officially contracted can submit claims.
* Each lecturer is linked to one Programme Coordinator and one Academic Manager.
* File uploads are limited to common formats like **.pdf, .docx, and .xlsx**.
* Claims are made once a month, with one claim allowed per lecturer.

**Constraints**

* The prototype must be built using **.NET Core MVC**.
* At this stage, it’s just a front-end prototype with no real functionality.
* Payroll and finance integration are not included in Part 1.
* The project must use version control with at least five meaningful commits.

**Conclusion**

This prototype sets the groundwork for the full system. It shows how the process will look and feel, without adding the actual logic yet. Later parts of the project will bring in functionality, automation, and extra features, but for now the focus is on having a clear design that reflects how the real workflow will operate.

A diagram of a company

AI-generated content may be incorrect.

| **Phase** | **Duration** | **Dependency** |
| --- | --- | --- |
| Requirements Gathering | 2 days | None |
| Database Design (UML) | 3 days | Requirements |
| GUI Prototype (WPF Tabs) | 5 days | Database |
| Documentation Report | 3 days | GUI |
| Testing & Refinement | 2 days | GUI, Docs |
| GitHub Commits (x5) | Continuous | Ongoing |

**Version Control (Commit Messages)**

1. Initial project setup with views
2. Added UML class diagram and database design draft
3. Implemented GUI prototype with Better Views
4. Updated views
5. Finalized documentation and updated project plan